SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

Faculty of Informatics and Information Technologies

FIIT-16768-112294

Olivér Izsák

Virtual Switch Implementation

PSIP Semester Project

Study program: Informatics

Semester: 6

Course: Switching and Routing in IP networks

Instructor: Ing. Ladislav Zemko

Date: March 9 2023

**Contents**

[Zadanie 3](#_Toc129551291)

[Implementation proposal 5](#_Toc129551292)

[1. Mac address and forwarding communication 5](#_Toc129551293)

[2. Threads and timers 6](#_Toc129551294)

[3. Physical Interference handling 6](#_Toc129551295)

[4. Statistics and filtering 7](#_Toc129551296)

[5.System logging 7](#_Toc129551297)

[Flowchart for the switch (excluding GUI and timers): 9](#_Toc129551298)

[References 10](#_Toc129551299)

# Zadanie

Navrhnite a implementujte softvérový viacvrstvový prepínač na základe znalostí získaných zpredmetu Počítačové a komunikačné siete (PKS). Pri spracovaní koncepcie návrhu prepínača uvažujte viacportový prepínač–implementovaná aplikačná logika by mala umožniť použitie N portov. Ako výsledná implementácia postačuje riešenie s dvojportovým prepínačom(dve sieťové karty, port 1 a port 2), pričom ovládanie sieťových rozhraní realizujte príslušnými paketovými ovládačmi. Prepínač navrhnite a implementujte v jazyku C++alebo C#(ďalšími povolenými jazykmi sú Java alebo Python).Navrhnite prepínač tak, aby spĺňal požiadavkyz úloh 1-5.

**Úloha 1: Prepínacia tabuľka**

Zobrazoval prepínaciu tabuľkuvo formáte MAC adresa–číslo portu–aktuálny časovač záznamu. Prepínač sa obsah svojej prepínacej tabuľky učí priebežne a aktuálny stav zobrazuje cez grafické používateľskérozhranie (obsah sa automaticky aktualizuje, nie pomocou tlačidla). Umožnite vyčistiť prepínaciu tabuľkupomocou tlačidla. Časovač pre vypršanie záznamov nech je konfigurovateľný (pozn.: nezabudnite ošetriť vytiahnutie kábla, ako aj výmenu káblov medzi portami–napr. pomocou monitorovania premávky na základe staticky definovaných MAC / IP adries, spravidla nekonečný ping medzi 2 zariadeniami).

**Úloha 2: Štatistiky**   
  
poskytoval štatistické informácie vrstvy 2-4 RM OSI o počte (prijatých/odoslaných) PDU na každom porte v smere INaj OUT, ktoré budú zreteľne zobrazovať správne fungovanie prepínača. Umožnite resetovať štatistické informácie prostredníctvom GUI. Štatistické informácie nech zobrazujú minimálne informácie o PDU typu Ethernet II, ARP, IP, TCP, UDP, ICMP, HTTP.

**Úloha 3: Filtrácia komunikácie**

Filtroval komunikáciu na 2.-4.vrstve RM OSI vrátane portov transportnej vrstvya typov ICMP(bez použitia vstavaných PCAP funkcií filtrovania). Riešenie navrhnite ako zoznam pravidiel vyhodnocovaných sekvenčne tak, aby bolo možné naraz realizovať ľubovoľnú kombináciu filtrov. Napr. pre danú IP povoliť iba HTTP komunikáciu a zároveň pre danú MAC zakázať "ping". Umožnite aj kombináciu zdrojových a cieľových MAC aIP adries, príp. portov. Zobrazujte tabuľku zadaných pravidiela umožnite ich aj jednotlivo odstraňovať. Filtre rozlišujte v smere "in/out"na každom porte prepínača(takisto zohľadniť vnávrhu). Napr. Host A sa nedostane von na web (HTTP), ale u neho bežiaci servernginx(HTTP) bude dostupný.

**Úloha 4: System Logging (Syslog)**

Implementácia Syslog klienta, pričom je potrebné:   
1.Zabezpečiť aspoň 3 úrovne dôležitosti správ (severity level).  
2.Umožniť nakonfigurovať prepínaču zdrojovú IP adresu, zktorej sa budú správy odosielať.  
3.NakonfigurovaťIP adresu vzdialeného Syslog servera.  
4.Zasielané správy musia obsahovať časovú pečiatku(angl. timestamp).  
5.Zvoľte aspoň 5 činností(descriptions), ktoré budete pomocou Syslog zaznamenávať(napr. „Zariadenie s MAC X sa premiestnilo z portu 1 na port 2“).  
Syslog server bude aplikácia TFTPD32 bežiaca na niektorom počítači(prípadne Networkers' Toolkitpre GNS3). Umožnite spustenie/zastavenie Syslog funkcionality na prepínači.

**Dokumentácia**

Dokumentácia musí obsahovať:  
 1.Zadanie úlohy.  
 2.Návrh riešenia obsahujúci podrobné diagramy spracovania rámcov s opisom čo sa kde a ako bude vykonávať(úlohy 1-3).  
 3.Analýzuprotokolov CDP alebo Syslog(implementácia bez dostatočnej analýzy nebude hodnotená), ak sa rozhodnete implementovať úlohu   
4.Dokumentáciu ako aj výsledný prepínač musí študent odovzdať do príslušného miesta odovzdania v AIS (po vložení súborov nezabudnúť súbory odoslať/odovzdať).

Všetky termíny určené miestom odovzdania v AIS sú konečné a za neskoré odovzdanie bude študent hodnotený 0b.  
Hodnoteniezadania Zadanie sa prezentuje a hodnotí priebežne po častiach, podľa pokynov cvičiaceho. Za oneskorené odovzdanie(t.j. študent nestihne do daného cvičenia/týždňa vypracovať určenú časť zadania) bude študent hodnotený 0b z príslušnej časti zadania. Predbežný plán odovzdávania a bodovania zadania:•3. cvičenie (3b): prototyp, ktorý musí vedieť prijímať a posielať komunikáciu (odchytiť prichádzajúci rámec na porte a poslať rámec von portom) + štatistiky. koniec 4. týždňa (2b+ 1b): dokumentácia(max 2bza úlohy 1-3,1b za úlohu 4).•7. cvičenie (10b): základná funkcionalita prepínača(úlohy 1-3).•koniec 10. týždňa (9b): filtre(4b) + CDP alebo Syslog(5b-len v prípade splnenia všetkých podmienok uvedených v zadaní, inak 0b).

# **Implementation proposal**

The goal of this assignment to create a multi-port virtual switch using 2 network adapters as ports.  
This implementation will be done in the programming language Java, because it makes working with Threads a lot easier, and there is no need to work with time consuming tasks such as freeing memory, allocating memory, pointers etc., since Java automatically takes care of these issues.

The implementation proposal for this assignment’s multi-port virtual switch will contain 5 main problems that need to be defined and described:

1. **Mac address and port-forwarding**
2. **Threads and timers**
3. **Physical interference handling**
4. **Statistics and filtering**
5. **System logging**

# **1. Mac address and forwarding communication**

The most vital part of any switch is the mac table. Which is responsible for storing data(MAC address, timer, port) of the devices that communicate through the switch.   
The MAC address is a 6 byte hexadecimal number that represents and identifies a specific device.  
There can be 0 or more MAC addresses associated with one switch port.  
The timer is responsible for making sure the limited memory of the switch doesn’t get full, so every switch has a timer which will remove entries that are outdated.  
The network adapters used by the program will be pretending to be the switch’s ports.

When a network adapter receives a packet first it will immediately check if the packet comes from another network adapter which is used by the virtual switch[[1]](#footnote-1). If yes, then it will throw away this packet. If the source MAC address is not from another network adapter and the MAC table doesn’t have this MAC address already, then it will write the source MAC address of the sender into the MAC table for the port it came in, and will start a 15 second timer. If a new packet comes with a MAC address that is already in the MAC table, then it will just update the entry.  
In the next action of the switch will be to change the source MAC address of the packet into the Network adapter’s MAC address. After that it will check the destination MAC address to decide where to send the packet. If the destination Mac address doesn’t match with any other MAC address in the MAC table, then it will forward it to all the other ports except the one it came in. If there is a matching MAC address it will only forward it to the specific port.  
In this implementation the mac table consist of 2 classes. 1 is the SwitchMacTable class. Which is a singleton object that means it has only one instance that is being accessed by all the other objects and threads. This class also contains a hash map for keeping track of the different MAC addresses to the corresponding port. The second class that is associated with the MAC table is the MacTableEntry class, which simply contains 3 objects the MAC address, timer and port.

# **2. Threads and timers**

This implementation requires a great number of timers and loops which means threads are needed.  
Firstly, there has to be a thread for each instance of the network adapter that is listening in a loop. Whenever one of the loops that are listening want to add a new mac address to the table they have to get the MAC table instance and wait for access so they can modify it. Thanks to the Java’s synchronized method specification the reader and writer problem can be easily solved, since the synchronized method automatically deals with locking the MAC table when it is being modified by another thread, and unlocks it when it is done. The same locking mechanism are used for the timers and GUI updates.

Timers are mainly used for 2 things. The first ones are the timers in the MAC table. Every time an entry is added to the MAC table, it will start a new Thread and a timer in it. Whenever a timer reaches a specific amount, the entry will be removed, and the GUI updated. The situation when the time is modified while the entry timers are running is dealt with the following way. First the new timer is calculated for each entry. If it is lesser than the current time the entry will be removed instantly. If it is not below it will get updated to the new time limit.

The second situation where timers and threads are needed are the ping’s which are for physical interference simulation and reaction. Which will be further explained in the 3. Point of this documentation.

# **3. Physical Interference handling**

This part includes the virtual representation of the scenarios when a cable is removed from the port of a switch and how the software handles it, and also the situation where the devices connected to the ports are switched.

First scenario uses a heartbeat ping method. This method uses ICMP packets to check if the connection to the device is still up. In the implementation this will start as soon as a MAC address is added to the MAC table. A new thread will start which will be pinging the connected device. After the first ping happens, it will wait 5 seconds for the connected device to reply. If it replies, the Thread will wait 5 seconds then it will start pining again. If the other device doesn’t reply in 5 seconds to the ping, then the device will simply be removed from the MAC table and the thread will stop.

The second scenario where the connected devices are switched from port1 to port2 can only happen if the switch happens in less than 5 seconds. Whenever this situation happens the MAC address entries will be deleted for the 2 devices, and new ones added with the updated ports. The way the program recognizes that the connected devices have changed their ports, is by pinging every 3 seconds every host that is connected to the switch and waiting for a reply in 2 seconds. If por1 is connected to pc1 and port2 is connected to pc2. Then both port1 and port2 will be pinging pc1 and pc2, and whenever the cables between pc1 and pc2 are switched to the other ports, then the original ping from port1 to pc1 will be unreachable, but the ping to pc2 will suddenly become reachable, which means we know that pc1 port has been switched in to port2.

# **4. Statistics and filtering**

The statistics are part of a Packet class which contains all the mentioned PDU types as an integer variable. Each port has 2 Packet classes, one is named OUT, the second is called IN. The IN packet class will get all the packets that the network adapter that is associated with the port listens to. Meanwhile the OUT packet class will contain the packets that are sent out from other network adapters to this one.

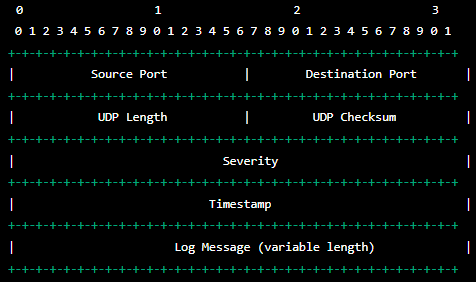
Each packet class can categorize the received packet by reading its bytes and checking it with if statements to check the PDU types and incrementing the corresponding variables. After a packet is categorized the GUI is updated.

The implementation will accept the following attributes when defining a filter:   
Direction (in/out)  
Source MAC address  
Destination MAC address  
Source IP address  
Destination IP address  
Transport layer protocol (TCP, UDP, ICMP)  
Transport layer source port  
Transport layer destination port  
Action (allow/deny)

For every port there will be a filter that can be configured with the attributes. After configuration, every packet that comes through that port first will check if it is going in or out. Next it will check if it contains the source/destination Mac or IP addresses, after that it will check the transport layer protocols, then the destination and source ports and finally whether it should allow or deny the packet. It also works in a way where if an attribute is enabled ,then that also means any other type is denied. Meanwhile every deny means allowing any other attribute. Each rule that is added to the rule list of a specific port will be read in a sequence from top to bottom. So the first rule will always be checked first, then the a second and so on. If there is a deny match for any packet in the list of rules, then it will stop there and will not go further down the rule list. Different rules will apply for on each port and for each Incoming and for Outgoing packets.

# **5.System logging**

The system logging will use a remote server running the TFTPD32 application. The ip address of both the switch and the remote server can be added by the user and the logging can also be turned on and off.  
There will be 5 actions that will logged each one having a severity level associated with it:  
  
1. Switch started or stopped. – Severity LvL 3.  
2. New MAC entry – Severity LvL 2.  
3. Mac entry removed – Severity LvL 2.  
4. Cable unplugged – Severity LvL 3.  
5. Filter configured – Severity lvl 1.

For the messages sent to the Syslog server UDP will be used, the header will look like this :   
  


Additional information for the header is that the source and destination ports will be 514.  
The entire packet will be minimum 48 bytes containing the Ethernet header(14) + ip(20) and udp(8) headers = 42 bytes an the added syslog information severity 2 bytes, timestamp 4 bytes, and finally the Log message can be between 0 – 1452.

# **Flowchart for the switch (excluding GUI and timers):**

# **References**

1. <https://www.cisco.com/c/en/us/support/docs/security/ios-firewall/23602-confaccesslists.html#anc11>
2. <https://en.wikipedia.org/wiki/User_Datagram_Protocol>
3. <http://philippe.jounin.pagesperso-orange.fr/tftp-server-for-windows.html>
4. <https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipswitch_poview/configuration/15-s/isw-poview-15-s-book.html>

1. The reason is because the way the pcap libraries sendPacket method works. Whenever that sendPacket method is called on a network adapter that is listening , it will automatically send it out on that network adapter, but also sends it to the other network adapters, which can cause a loop that will flood both ports. [↑](#footnote-ref-1)